eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

Trabajo Fin de Grado
Grado en Ingeniería Electrónica

Control de seguimiento de un vehículo mediante tecnologías inalámbricas

Soportado en la placa de desarrollo Arduino

Autora:
Marta Hernández Aycart
Director:
Javier Echanobe

Leioa, 19 de junio de 2020

Gradu Amaierako Lana/Trabajo Fin de Grado

Ingeniaritza Elektronikoko Gradua/Grado en Ingeniería Electrónica

---

Control de seguimiento de un vehículo mediante tecnologías inalámbricas

---

Marta Hernández Aycart

Director:

Javier Echanobe

Facultad de Ciencia y Tecnología
Universidad del País Vasco UPV/EHU

Leioa, Junio 2020

Índice general

Capítulo 1

Introducción y objetivos

El objetivo último de este trabajo es el control de movimiento de un vehículo soportado por la plataforma de desarrollo de código abierto Arduino, tanto mediante el uso de un sensor basado en ondas de ultrasonido como el control remoto mediante tecnologías infrarrojas. Para conseguir el objetivo principal, se han propuesto, a su vez, una serie de objetivos específicos:

- Estudio y análisis de la plataforma Arduino y su entorno de programación
- Caracterización de sensores basados en ondas de ultrasonido
- Montaje del vehículo y conexionado de los motores
- Conexionado del sensor de distancia y sensores infrarrojos a la placa Arduino
- Tests para la verificación del funcionamiento de los programas implementados

El trabajo está estructurado en 6 capítulos. El primero es el presente capítulo introductorio. En el segundo de ellos se realiza un estudio de la plataforma de desarrollo Arduino, haciendo un pequeño análisis de su arquitectura, funcionamiento y entorno de programación.

En el tercer capítulo se describen los sensores utilizados. Además, se realiza la caracterización de dos tipos de dispositivos sensores de ultrasonidos para escoger el más adecuado para el proyecto. En el capítulo cuarto, se presenta los componentes mecánicos que se van a utilizar: motores DC y servomotor. Para ambos casos, se realiza un estudio de sus características principales y su modo de uso.

En el capítulo 5, se exponen las dos aplicaciones desarrolladas. En primer lugar se implementa un control automático, en el que el vehículo se mueve con libertad esquivando obstáculos gracias al uso de un sensor de proximidad. En segundo lugar, se implementa el control del vehículo mediante la tecnología inalámbrica infrarroja. Además, se proponen mejoras y alternativas a implementar haciendo uso del mismo vehículo.

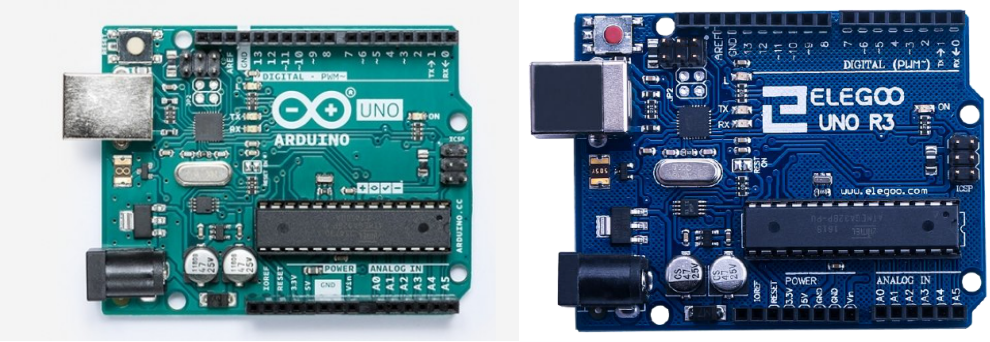
En el último capítulo se exponen las conclusiones obtenidas a lo largo del trabajo, así como se presentan aspectos más personales adquiridos en el proceso de aprendizaje.

Capítulo 2

Descripción de la plataforma

Para el control de movimiento de los motores y la comunicación entre estos y los sensores, es necesario el uso de un microcontrolador. Un microcontrolador es una pequeña computadora que realiza tareas específicas. En este caso, se ha escogido un microcontrolador de la familia Arduino, puesto que son muy sencillos de programar, son unas plataformas de precio muy asequible y existe infinidad de bibliografía al respecto, por lo que su uso es muy sencillo.

Al hablar de "Arduino", la mayoría piensa en él como la pequeña, rectangular y azul PCB[1] (figura 2.1a). Realmente, esta tarjeta es la tarjeta de entradas-salidas (*I/O Board*), y se conoce "Arduino" como el conjunto de hardware, software, equipo de desarrollo, filosofía de diseño y *esprit de corps* [8].



(a) Placa Arduino UNO

(b) Placa Elegoo UNO

Figura 2.1

Se creó originalmente en Italia en el año 2005, cuando todavía era muy costosa la adquisición de microcontroladores. Por ello, el entonces estudiante Massimo Banzi decidió crear esta plataforma para el aprendizaje de los estudiantes de computación y electrónica del instituto Ivrae, donde se construyó el primer prototipo. Inicialmente, éste consistía únicamente en una placa de circuitos conectada a un microncontrolador a la que podían conectarse sensores simples y que no tenía soporte de ningún lenguaje de programación.

Con el tiempo, estudiantes de computación y electrónica de diferentes universidades del mundo se fueron interesando por el proyecto y aportando sus mejoras: el desarrollo del entorno de programación, mejorando la interfaz del hardware añadiéndole memoria al lenguaje, añadiendo puertos USB para la conexión con un ordenador, etc. A día de hoy se han vendido más de 250 mil placas,

[1] PCB: *Printed Circuit Board*. Tarjeta de circuito impreso

sin contar con las versiones clones [8].

Al hilo de esto último, cabe mencionar que los creadores de Arduino hicieron de éste un *open hardware*. Esto implica que compartieron toda la información sobre el código fuente, documentación técnica y esquemáticos. Se podría, por tanto, construir una placa idéntica a las desarrolladas por Arduino si tuviéramos todos los recursos necesarios para ello. Hay muchos fabricantes que comercializan copias y uno de los más conocidos, y el que se va a utilizar para el proyecto, es ELEGOO (figura 2.1b). En concreto, se va a utilizar el modelo UNO, que cuenta con el procesador ATMega 328: procesador de 8 bits, 16MHz de frecuencia de reloj y 32KB de memoria FLASH .

En el siguiente punto, se van a explicar los componentes de la placa Arduino UNO.

2.1. Arquitectura del hardware

En la figura 2.2, están señaladas las partes que componen la placa del Arduino Uno. Por un lado, se pueden diferenciar los elementos electrónicos embebidos y, por otro, los pines. A continuación, se listarán todos los componentes y se hará una breve descripción de ellos:

(1): Conector USB: este conector sirve, por un lado, para alimentar la placa y, por otro, para subir el código desde el ordenador. Una vez que éste está subido, se puede alimentar la placa con el adaptador de alimentación **(2)** (6-12V).

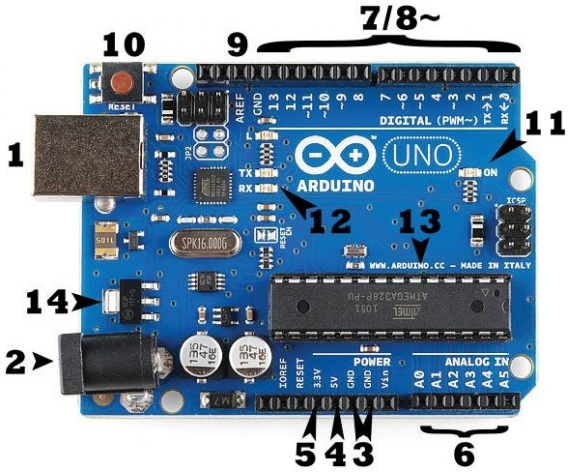


Figura 2.2: Placa Arduino UNO

Los pines de tierra GND **(3)**. La placa cuenta con 3 pines, cualquier de los cuales pueden ser utilizados ya que se trata realmente del mismo punto. Los pines **(4)** y **(5)**, son los pines de suministro de tensión de 5V y 3.3V, respectivamente.

Situados a la derecha de los anteriores pines, se encuentran los pines de entradas analógicas **(6)**. Esta placa cuenta con 6 pines que pueden leer señales de sensores analógicos y convertirlos en valores digitales para su visualización.

La placa cuenta también con 14 pines digitales **(7)**, que se pueden usar tanto de entrada como de salida. Además, algunos de estos pines son los PWM[2] **(8)**, tildados con el símbolo (∼). En esta tarjeta, como se puede observar en la figura 2.2, los pines analógicos se pueden utilizar únicamente

[2]PWM: *Pulse Width Modulation*

como señales de entrada. Para simular una salida analógica se activa una salida digital durante un tiempo y se mantiene apagada durante el resto. De esta manera, haciendo el promedio de la tensión de salida, a lo largo del tiempo, se consigue el valor analógico deseado. Este es el caso de los pines PWM, la modulación se mantiene constante en frecuencia mientras se varía la anchura del pulso. Sin embargo, a la hora de utilizar estos pines es importante tener en cuenta que no son señales analógicas y que su valor de tensión realmente es Vcc. Por lo tanto, se ha de tener mucho cuidado con los dispositivos que se conectan a estos pines porque si, por ejemplo, se conecta un dispositivo que necesita 3V, realmente se está suministrando 5V durante el 60 % del tiempo y 0V durante el 40 % restante. Si dicho dispositivo, sin embargo, soporta como máximo 3V al alimentarlo mediante un PWM podría dañarse.

El pin AREF **(9)**, es un comparador de tensiones analógicas. Compara la tensión de referencia que se introduzca externamente a ese pin con las señales que se lean en las puertas analógicas A0-A5. Este pin es muy útil para aprovechar completamente la resolución del microcontrolador.

El microcontrolador ATmega 328 utiliza un conversor analógico-digital de 10 bits, lo que supone que se puede representar una señal analógica de $2^{10} = 1024$ valores. Por lo tanto, existen 1024 combinaciones de bits para representar una señal analógica que irá de 0 a 1023. Sin embargo, estos valores analógicos se toman por defecto en referencia a los 5V máximos que puede aportar la tarjeta; por lo tanto, esas 1024 combinaciones están separadas por escalones de 4.88mV. Cuando el voltaje en la entrada del conversor sea de entre 0 y 4.88mV, se obtendría un 0 digital, se dice entonces que la resolución del microcontrolador es de 4.88mV.

Se supone ahora que se está utilizando un sensor cuyo valor máximo de salida se conoce que es 1.5V (un sensor de temperatura LM35, por ejemplo). Sabiendo eso, de antemano, y para no desperdiciar la resolución se hace uso del pin AREF, introduciéndole una fuente de tensión del voltaje deseado (1.5 V), de manera que la resolución con la que se obtengan las señales pase de 4.88mV a 1.46mV.

A su vez, la placa cuenta con un botón de reinicio **(10)**, que resulta de utilidad cuando el código no se repite pero quiere probarse más de una vez. El indicador LED de alimentación **(11)** se enciende cuando la placa está conectada a una toma eléctrica, y de estarlo y no encenderse puede servir de indicador de que hay algo que no funciona como debiera.

En el mundo de la electrónica embebida el protocolo de comunicación serie asíncrono es muy utilizado y el dispositivo hardware que controla los puertos serie se llama UART[3], que en el caso de la placa Arduino UNO se encuentra conectado a los pines 0 y 1. La función de los UART es convertir los datos serie en paralelo cuando se trata de datos de entrada (recibidos) y hacer lo contrario cuando los datos son del tipo salida (transmitidos). Además del LED de alimentación, cuenta con otros LEDs **(12)**, RX/TX. Estos son, respectivamente, las abreviaturas de recibir y transmitir, y se utilizan para visualmente poder conocer en cada momento si la tarjeta está mandando o recibiendo datos.

La parte, quizás, más importante de la tarjeta es el microcontrolador **(13)**, que como se ha mencionado es el ATmega 328. Por último, cuenta con un regulador de tensión **(14)**. Con este elemento, realmente no se interactúa pero conviene conocer su importancia para la placa. Como su nombre indica, controla/regula la cantidad de voltaje que se aplica a la placa Arduino y no permite que pasen tensiones que puedan dañarla.

[3]UART: *Universal Asynchronous Receiver-Transmitter*. Receptor-Transmisor Asíncrono universal

2.2. Descripción del entorno de programación Arduino

Un entorno de programación es el programa o conjunto de programas que incluyen las tareas necesarias para la programación en dicho lenguaje. Estas tareas son: la edición del programa, la compilación, la ejecución y la depuración de errores. A los entornos de programación se les llama comúnmente IDE[4] . El IDE del Arduino, cuenta con un editor de texto para escribir código en lenguaje C++, un área para mostrar mensajes, una barra de herramientas y una consola de texto.

Además, este software ofrece una biblioteca de ejemplos básicos sobre los que ir desarrollando el programa. Una vez está listo el código (o *sketch*)que se desea subir a la placa, el software ofrece las herramientas necesarias para compilarlo y cargarlo en la memoria flash del dispositivo a través del puerto serie.

En la figura 2.3, se puede observar una captura de pantalla de la ventana que se abre nada más iniciar el IDE. Como se puede observar, cuenta con dos partes que todo programa Arduino debe tener: *setup* y *loop*. En el primero, se deben incluir la inicialización de todos los pines de la tarjeta, la declaración de variables y se configura la comunicación con el equipo. Esta función se ejecuta una sola vez. La función *loop* se ejecuta inmediatamente después del *setup* y debe incluir el programa que se desea ejecutar en bucle. Además, antes de estas dos funciones es conveniente realizar la adjudicación de pines de la placa, es decir, indicar cada pin a qué componente del circuito está conectado. De esta manera, se facilita la programación [6].

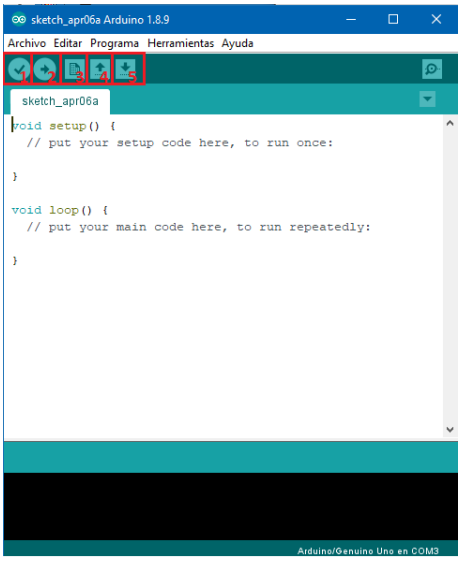


Figura 2.3: Captura de pantalla IDE

Los botones marcados en la figura 2.3 corresponden a lo siguiente:

1. Verificar/Compilar: antes de subir el *sketch* a la placa Arduino, se debe comprobar que no haya errores.
2. Subir: carga el *sketch* a la placa.
3. Nuevo: abre una ventana nueva como la de la figura 2.3
4. Abrir: carga un *sketch* guardado en el PC.

[4]IDE: *Integrated Development Environment*. Entorno integrado de programación

5. Salvar: guarda el programa.

A su vez, dispone de una barra con diferentes pestañas. Las más interesantes, probablemente, sean la pestaña de *Programa* y la pestaña de *Herramientas*. En la primera, se puede encontrar la opción de incluir biblioteca. Por defecto, al instalar el software del entorno IDE, se instalan una serie de librerías básicas que incluyen la declaración de las funciones necesarias para la utilización de una gran cantidad de componentes. Por ejemplo, para la utilización de la pantalla de cristal líquido (LCD[5]) se debe incluir la biblioteca < *"LiquidCrystal.h"*>, y para ello hay que importarla previamente al programa (incluir el *path* donde se encuentra). Además, esta pestaña incluye también la opción de *Añadir fichero*. Esta opción puede resultar de gran utilidad, cuando se quiere realizar un proyecto que incluya diferentes componentes, cada uno de los cuales requiere un programa que lo controle. De esta manera, se puede tener un programa principal, que a su vez, vaya llamando a diferentes programas que realizan diferentes funciones. Esto facilita enormemente la detección de posibles errores y la comprensión del proyecto, ya que éste se encuentra estructurado por bloques.

En la pestaña *Herramientas*, se encuentran las opciones de control de la placa: puerto en el que se encuentra conectada al PC, tipo de comunicación con éste y toda la información referente al modelo. Además, desde esta pestaña se puede abrir el *Monitor Serie*, que permite escribir mensajes por pantalla o el *Serial Plotter*, que permite fácilmente graficar.

2.3. Lenguaje de programación

Como se ha mencionado, el lenguaje de programación de Arduino está basado en C++. Este lenguaje se creó con la intención de extender el exitoso lenguaje C para poder incluir la posibilidad de definir y manipular clases y objetos, por eso en cuanto a la orientación a objetos se dice de C++ que es un lenguaje híbrido [6].

Se trata de un lenguaje con pocas restricciones sintácticas y bastante intuitivo. Sin embargo, a la hora de programar con él, se han de tener en cuenta varios detalles sin los cuales el compilador genera un mensaje de error. En primer lugar, el uso de las llaves ({}), todo lo que va entre ellas define un bloque de instrucciones. Si una llave de apertura no trae consigo una de cierre, el compilador emite un mensaje de error. Para evitarlos, el entorno de programación incluye la opción de posicionarse encima de una llave de apertura o cierre, y éste marca la posición de la otra. En segundo lugar, todas las instrucciones en el lenguaje deben ir separadas por un punto y coma (;), de no ser así, de nuevo, el compilador daría error.

Además, como se ha mencionado anteriormente, es muy sencillo encontrar en la red gran cantidad de foros y páginas donde, no solo se pueden resolver dudas, sino que se pueden encontrar *sketchs* enteros.

[5]LCD: *Liquid-Crystal Display*

Capítulo 3

Descripción de los sensores utilizados

En este capítulo se van a definir los diferentes sensores que se utilizarán para las aplicaciones. En primer lugar, un sensor de ultrasonidos para calcular la distancia a la que se encuentra un objeto situado en la dirección del mismo. En segundo lugar, se utilizarán también dos tipos de sensores que hacen uso de tecnología infrarroja. Uno de ellos se utilizará para la detección del suelo y otro para el control del movimiento del vehículo.

3.1. Sensores de ultrasonidos

Quizá el componente más importante del trabajo es el sensor de ultrasonidos que se va a utilizar. Se trata de un sensor basado en la tecnología de ondas ultrasónicas, ondas de frecuencias comprendidas entre 20kHz y 100kHz. El dispositivo manda una ráfaga corta ultrasónica, que viaja a través del aire hasta "chocar" contra un objeto, donde rebota. El sensor recibe la onda rebotada (el eco) y calcula la distancia estudiando el tiempo entre la emisión de la ráfaga y la detección del eco.

El sensor está construido por un material piezoeléctrico. Estos materiales sufren el denominado "efecto piezoeléctrico", por el que manifiestan cargas, corrientes o diferencias de potencial al someterlo a un esfuerzo mecánico (piezoeléctrico directo) o bien, sufren alguna deformación al someterlo a una diferencia de potencial (piezoeléctrico inverso). En este caso, el material del que está formado el sensor es un piezoeléctrico inverso, al que se le aplica una corriente, que hace que los cristales piezoeléctricos del material oscilen a la misma frecuencia produciendo ultrasonidos.

El dispositivo que se va a utilizar es un transceptor ultrasónico, lo que significa que contiene tanto el emisor, que convierte las señales eléctricas en ondas de ultrasonidos, como el receptor, que hace lo contrario.

La distancia a la que se encuentra el objeto se puede calcular fácilmente conociendo la velocidad a la que viaja el pulso (340 m/s) y el tiempo que tarda éste en ir y volver:

$$d = v \cdot \frac{t}{2} \tag{3.1}$$

El protocolo de comunicación entre el sensor y la tarjeta Arduino es muy sencillo y no requiere de ningún otro dispositivo intermedio para llevarla a cabo. El pin SIG ha de conectarse a una entrada digital de la placa. Se escribe un '1' lógico en él y se espera al regreso de la señal, se mide el tiempo que tarda la señal en ir y volver y se traduce en distancia a la que se encuentra el objeto.

El detector más utilizado como sensor de distancias en proyectos de robótica es el sensor de ultrasonidos HC-SR04 (figura 3.1a) que se comercializa normalmente en los kits de Arduino y suele ser suficiente para pequeños proyectos. Sin embargo, en esta ocasión se ha adquirido también el dispositivo PING))) #28015 del fabricante Parallax (imagen 3.1b) y se van a realizar unas pruebas para caracterizarlos, obtener las principales diferencias y concluir cuál es más apropiado para este proyecto. Ambos cuentan con dos pines de alimentación (Vcc y Tierra) y otro (u otros dos) encargados de mandar y recibir el eco ultrasónico. Ambos dispositivos incluyen toda la electrónica necesaria integrada en la misma placa.

(a) Sensor HC-SR04 (b) Sensor PING))) #28015

Figura 3.1: Sensores de distancia ultrasónicos

En la tabla 3.1, se han recogido las principales características de ambos dispositivos.

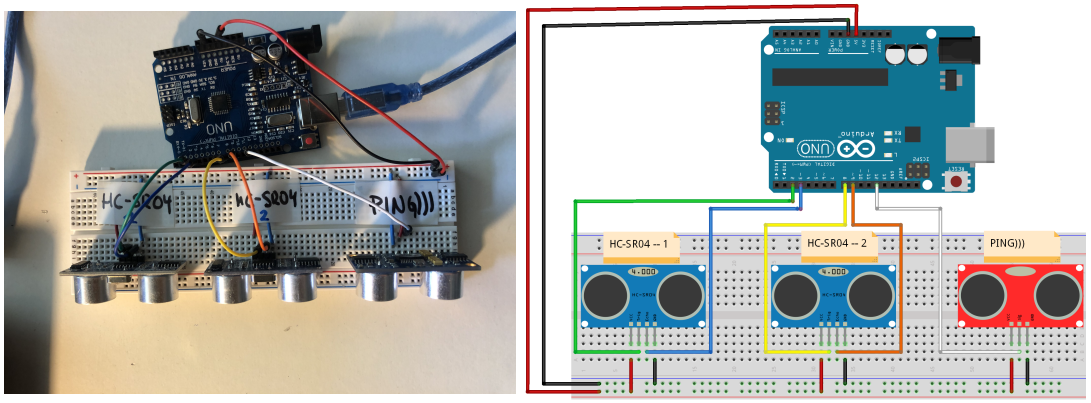
	PING)))	HC-SR04
Dimensiones	(22 x 46 x 16) mm	(43 x 20 x 17) mm
Peso	9 g	10 g
Ángulo de trabajo	$< 45^o$	$\pm 15^o$
Voltaje de alimentación	Vdd = 5V	Vdd = 5V
Corriente de alimentación	Típ: 30 mA (máx 35mA)	15mA
Rango de medida	(2.00 – 300.00) cm	(2.0 – 400.0) cm
Frecuencia de trabajo	40 kHz	40 kHz
Comunicación	Estándar TTL	Estándar TTL

Tabla 3.1: Comparativa sensores de ultrasonido [2] [3]

Como se puede observar en la figura 3.1, el dispositivo HC-SR04 cuenta con un pin más ya que con uno se manda la onda y con otro se recoge el eco, cosa que puede resultar conflictiva ya que hay que utilizar un pin más de la placa Arduino. Por lo demás, su funcionamiento es muy parecido. Las dos grandes diferencias, no obstante, que presentan son, por un lado, el coste y, por otro, la sensibilidad que tienen. Para esto último, se ha hecho una comparativa con dos sensores HC-SR04 y el dispositivo PING))), midiendo a la vez y comparando resultados.

Para ello, se ha montado un pequeño circuito (figura 3.2) en el que se han conectado los 3 dispositivos y se han hecho medidas colocando un objeto a 8cm, 15cm, 20cm, 25cm, 30cm y 40cm. El motivo de haber utilizado dos dispositivos "iguales" se ha debido a que en cierta bibliografía de proyectos con Arduino se demostró que había gran diferencia de un dispositivo a otro pese a tratarse del mismo modelo. Por ello, y ya que se contaba con dos dispositivos del mismo modelo, se decidió comprobar la diferencia de medidas.

Además, en la figura 3.3, se puede observar una foto de cómo se hicieron las pruebas de caracterización. Se colocaron cintas marcando la distancia y sobre la parte más próxima a la protoboard (correspondiente a la distancia que marca), se colocó un objeto que ocupaba todo el espacio de trabajo de los sensores.



(a) Foto del circuito

(b) Esquemático del circuito

Figura 3.2: Circuito para la caracterización de los sensores de distancia

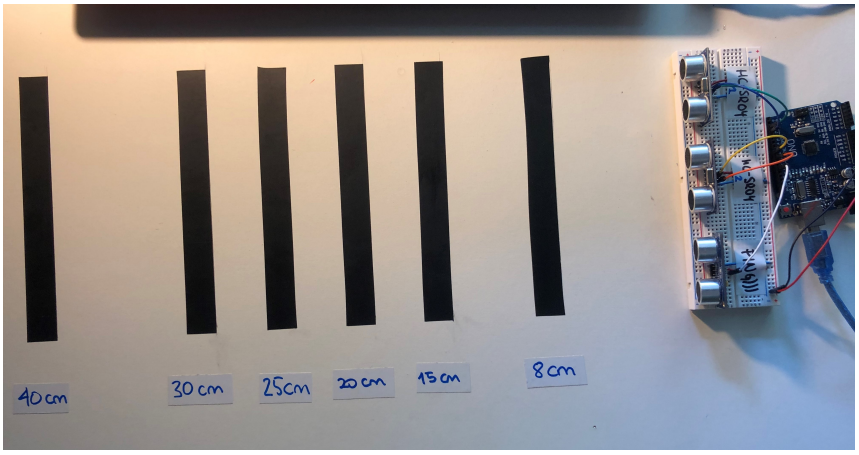


Figura 3.3: Montaje para las pruebas de caracterización

Se hizo un estudio de la repetibilidad (toma de medidas en las mismas condiciones para la caracterización de los sensores), haciendo 5 medidas con cada sensor para cada distancia. En la tabla 3.2, se han recopilado las medias para cada sensor. Como se puede observar por los datos obtenidos, los dos sensores del mismo modelo presentan resultados muy diferentes, obteniéndose con uno de ellos en todas las ocasiones valores muy próximos al real y con el otro, sin embargo, valores siempre menores.

Distancia objeto	**Valores medidos**		
	HC-SR04 – 1	HC-SR04 – 2	PING)))
8 cm	8.8	6.6	8.52
15 cm	15.2	13.2	15.75
20 cm	19.8	18.6	20.08
25 cm	24.2	23.0	25.89
30 cm	29.8	27.8	29.99
40 cm	39.2	32.6	38.38

Tabla 3.2: Medias de las medidas tomadas con los diferentes sensores de ultrasonidos

En cuanto al sensor de Parallax, en todas las ocasiones se obtienen valores parecidos al real. Sin embargo, las diferencias entre éste y el HC-SR04-1 no permiten concluir categóricamente que el primero sea mejor. Por lo tanto, ya que no se puede elegir uno de ellos por su precisión a la hora

de medir, se podría hacer prestando atención al precio. El coste del sensor HC-SR04 no llega al euro, además es un dispositivo que viene incluido en la gran mayoría de *kits* de iniciación de Arduino, Elegoo y con los *kits* preparados para pequeños proyectos de robótica. El detector PING))) de Parallax, por su parte, tiene un coste de aproximadamente 35 €. Por ello, el uso del sensor PING))) para este proyecto no sería necesario; aunque, ya que se tiene se va a utilizar éste por el uso que hace de un pin menos de la placa Arduino.

El *sketch* utilizado para la toma de estas medidas, se adjunta en el Anexo A.1.1 y cabe destacar que se ha introducido un retardo entre las medidas entre los sensores debido a que en pruebas iniciales se observó que los resultados obtenidos haciendo pruebas simultáneas no eran coherentes.

3.2. Sensores infrarrojos

Estos sensores suelen estar normalmente compuestos por un LED infrarrojo y un fototransistor, uno junto al otro. De esta manera, el LED actúa como emisor de luz infrarroja y el fototransistor como receptor. La luz infrarroja pertenece a la parte del espectro electromagnético con longitudes de onda comprendidas entre 0.3 y 1000 μm, entre la luz visible y las microondas (figura 3.4), y por lo tanto se trata de una radiación invisible al ojo humano. Si esta luz incide en una superficie blanca, esta reflejará la luz llegando dicha onda reflejada al fototransistor. Si, sin embargo, incide en una superficie negra esta absorberá la mayoría de radiación impidiendo que llegue de vuelta al receptor (figura 3.5).

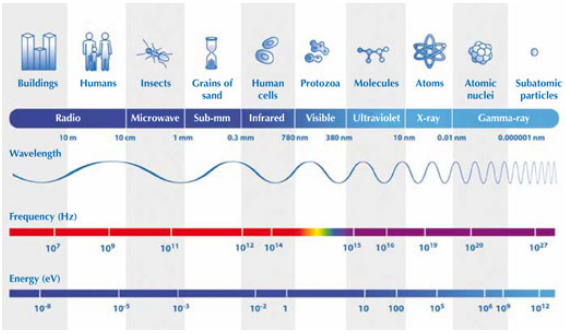


Figura 3.4: Espectro electromagnético

Este tipo de sensores pueden ser activos o pasivos. En el primer caso, el sensor está compuesto por un emisor de radiación infrarroja y detector de la misma; en el segundo, el sensor es básicamente el detector de energía emitida por los obstáculos que se encuentran en su campo de visión, sin utilizar ninguna fuente de radiación.

Además del sensor de ultrasonidos para el cálculo de la distancia a la que se encuentran los obstáculos en la dirección de movimiento del vehículo, se va a hacer uso de un sensor de infrarrojos para detectar si hay suelo por si, por ejemplo, se encontrase con un escalón.

Podría suponerse que este tipo de sensores se podría utilizar para medir distancias, sin embargo, no cuentan con una precisión suficiente como para proporcionar un valor de la distancia al obstáculo. Esto se debe a que la cantidad de luz infrarroja que recibe el fototransistor depende de factores como el color, la forma, el material y la posición a la que se encuentre el obstáculo. Además el campo de medida en el que actúan suele típicamente ser de entre 5 y 20 mm [5].

Por tratarse de un sensor que cuenta con componentes tan sencillos su montaje podría llevarse a cabo por nosotros mismos, sin embargo, en general no merecería la pena ya que la compra de los componentes supondría un coste superior a la compra del chip integrado (cuyo precio ronda los

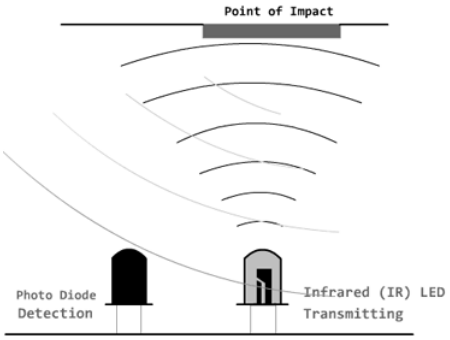
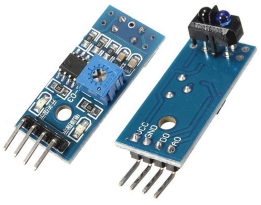


Figura 3.5: Funcionamiento detectores infrarrojos

0.3€).

Se dispone, para hacer este proyecto, de dos tipos de sensores infrarrojos (figuras 3.6a y 3.6b), ambos activos. Como se puede observar, uno de ellos tiene 3 pines y el otro 4. El primero tiene 2 pines para la alimentación (GND y 5V) y otro pin para la lectura digital del fototransistor. Además, cuenta con una pequeña placa comparadora, que acciona la lectura digital una vez se supera un cierto umbral, cuyo valor puede regularse manualmente con el potenciómetro integrado en el chip. Para calibrar el umbral de disparo, se acerca un objeto al detector y se va regulando la salida digital con el potenciómetro.

(a) Sensor IR FC-51



(b) Sensor IR TCRT-5000

Figura 3.6: Sensores infrarrojos activos

El segundo modelo, tiene 4 pines: los dos de la alimentación, uno para la lectura digital y el cuarto, y con el que no cuenta el primer modelo, lee una señal analógica cuya tensión varía cuando lo hace la distancia a la que se detecta el objeto. Por lo demás, también cuenta con la placa comparadora y el regulador.

Para este proyecto, se utilizará el segundo modelo aunque esta elección no se ha hecho por el cuarto pin con el que cuenta sino porque el LED y el fototransistor están dispuestos de manera que los rayos se dirigen en una dirección y, colocándolo hacia el suelo, no detectará la presencia del propio chasis, como podría pasar si se hace uso del otro modelo. En la tabla 3.3, se han recogido las principales característica del detector escogido.

Para comprobar el funcionamiento de este sensor, se ha montado un pequeño circuito en el que un LED rojo se encenderá si el sensor detecta un obstáculo y uno verde en caso contrario (figura 3.7). El código Arduino para controlar este circuito es muy sencillo y se puede encontrar en el Anexo A.1.1. El funcionamiento es muy sencillo: si llega luz al receptor (es decir, se ha encontrado un obstáculo en el que se ha reflejado la luz) se ilumina el LED de la placa y se manda una señal de LOW. Si, por el contrario, no llega luz (no ha habido reflexión), no se ilumina el LED de la placa y se manda un HIGH. En el primer caso, se escribe el LOW en el LED verde del circuito y se niega para escribir un HIGH en el LED rojo. En el segundo caso, se hace lo contrario. De esta

Voltaje de operación (V)	3.3 - 5.0
Corriente de operación (mA)	23 - 43
Angulo de detección	$\sim 35^o$
Distancia de detección (cm)	2 - 30
Dimensiones (cm)	3.2 x 1.4 x 0.7
Peso	3 g

Tabla 3.3: Características módulo TCRT-5000

manera se comprobó el correcto funcionamiento del detector.

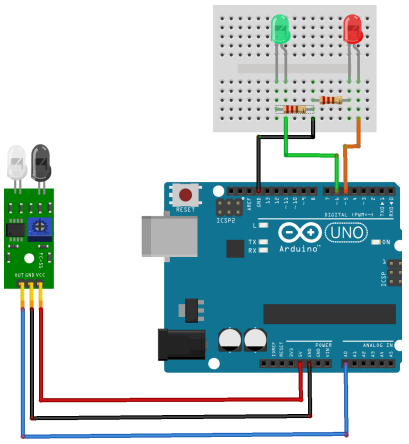


Figura 3.7: Esquema del circuito para comprobar el funcionamiento del detector infrarrojo

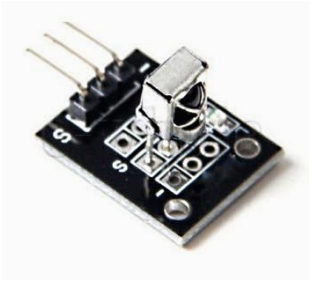
Además de estos dos sensores infrarrojos para detectar el suelo se va a utilizar otro par de componentes que hacen uso de la misma tecnología: mando a distancia y detector infrarrojos. En el caso anterior, el mismo encapsulado incluía emisor y detector de radiación infrarroja; en éste, sin embargo, el detector estará conectado a un pin del Arduino y el emisor estará encapsulado en un pequeño mando para poder enviar órdenes en remoto.

Este tipo de dispositivos están muy presentes hoy en día ya que se utilizan para el control de diferentes equipos electrónicos como puede ser una televisión o un reproductor de música. De hecho, los mismos mandos que se usan para dichos dispositivos podrían utilizarse para el control del Arduino. Sin embargo, en este caso se utilizará la pareja emisor-receptor HX1838 (figura 3.8a y 3.8b). Se ha escogido estos componentes por su bajo coste, ya que se puede encontrar la pareja por menos de un euro.

Para transmitir todo mensaje es necesario que tanto emisor como receptor respondan al mismo protocolo de comunicación. En el campo de los mandos a distancia infrarrojos, sin embargo, no existe un único protocolo adoptado como estándar y cada fabricante ha desarrollado el suyo propio. Uno de los protocolos más habituales, y el que se utilizará para el proyecto, es el protocolo NEC desarrollado por la compañía Nippon Eletronic Company. Sin embargo, también se pueden encontrar los protocolos RC-5 y RC-6 de Philips o el SIRC de Sony.

El mensaje no se envía únicamente como un pulso sino como una onda modulada de luz infrarroja. En el protocolo NEC se emplea una onda portadora de 38kHz y la modulación por distancia de pulsos (PDM[1], también conocido como *space encoding*). De esta forma, se codifica la señal modulando la separación entre pulsos mientras la onda se mantiene constante. Por ejemplo, un '0'

[1]PDM: *Pulse Distance Modulation*



(a)

(b)

Figura 3.8: Pareja emisor-receptor IR

lógico podría codificarse como un pulso de 565.5 μs seguido por 562.5 μs de espacio; y, un '1' lógico sería un pulso de 562.5 μs seguido por 1.6175 μs de espacio [7].

Haciendo uso del protocolo NEC, se envía una dirección de 8 bits y un comando de 8 bits, pudiendo así controlar 256 dispositivos diferentes sin interferir entre ellos y pudiendo enviar hasta 256 comandos distintos. Además, se envía el mensaje y su negación para evitar errores. Por ello, el mensaje total transmitido consta de:

- Pulso de 9 ms
- Espacio de 4.5 ms
- 8 bits: dirección del receptor
- 8 bits: dirección negada del receptor
- 8 bits: comando
- 8 bits: comando negado
- 562.5 μs de pulso para marcar el fin del mensaje

El receptor utilizado para esta aplicación es el AX-1838HS encapsulado en una pequeña placa con la electrónica integrada. Cuenta con tres terminales para la alimentación y para la señal de salida, que se conectará a un pin digital del Arduino. Para su programación, existen múltiples librerías pero para esta aplicación se utilizará la librería $<$*"IRremote.h"*$>$.

Haciendo uso de esta librería se puede trabajar con múltiples protocolos de comunicación. En primer lugar, se debe conocer los datos hexadecimales de cada botón del mando. Para decodificarlos la librería que se utiliza cuenta con una función llamada *decode_results*, donde se pueden diferenciar los diferentes protocolos e imprimir los comandos correspondientes a cada tecla por el puerto serial. Las órdenes que se envían tienen la siguiente estructura:

$$Dir(HEX) + \overline{Dir(HEX)} + Comando(HEX) + \overline{Comando(HEX)} = \ dato\ 32\ bits\ (HEX)$$

De esta manera, el dato 0x00FFA25D correspondiente al botón de encendido del mando utilizado con el protocolo NEC estaría formado por: 00/FF de la dirección y su negación; y A2/5D del comando y su negación. Por ello, todas las órdenes con las que se trabajará empezarán por 00FF ya que es la dirección del único mando del que se va a hacer uso.

En el Anexo A.1.2, se ha incluido el *sketch* utilizado para decodificar todo lo botones.

Cuando se conoce la codificación de cada botón, se realiza un pequeño circuito formado por cuatro LEDs: rojo, azul verde y amarillo, que se encenderán cuando se pulsen las teclas 1-4, respectivamente, y permanecerán encendidos 1 s. En la figura 3.9, se puede observar el esquema de conexiones y en el Anexo A.1.2 el código utilizado.

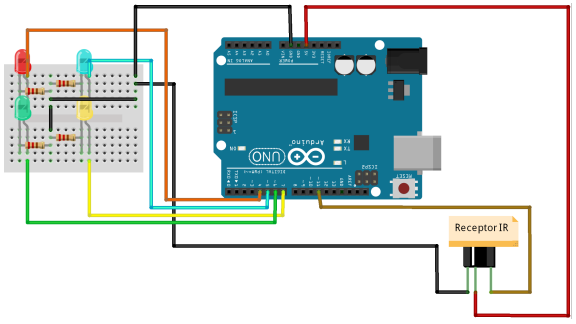


Figura 3.9: Esquema del circuito para comprobar el funcionamiento de la pareja emisor-detector IR HX1838

Capítulo 4

Descripción de los componentes mecánicos del vehículo

En este capítulo se exponen los dos motores de los que se va a hacer uso: motor DC para el movimiento del vehículo y servomotor para mover el sensor de ultrasonidos en diferentes direcciones.

4.1. Motor DC y su controlador

Como se ha mencionado, se va a hacer uso de dos parejas de motores de corriente continua (DC) para controlar las 4 ruedas del vehículo. Este tipo de motores convierte energía eléctrica, en este caso suministrada por unas pilas, en energía mecánica.

Este tipo de motores constituyen la aplicación más sencilla de los fundamentos de la máquina lineal en una máquina rotativa. Se componen, principalmente, de dos partes: el estátor y el rotor. En el primero, que no gira, se crea el campo magnético que hará que luego gire el rotor; puede estar compuesto bien por devanados de hilo de cobre, o bien por dos imanes fijos.

Aunque es posible el control de algunos motores de baja potencia directamente con la placa Arduino, cuando la potencia es grande se necesita hacer uso de un controlador de motor o *driver*, que proporcione la potencia necesaria. Además, para controlar el sentido de un motor de corriente continua, hace falta invertir la polaridad con la que se alimentan los bornes de éste y para ello habría que invertir la batería. Es evidente, que para proyectos de robótica no existe la posibilidad de invertir la batería cada vez que haya que cambiar el sentido de giro de los motores. Por ello, se utiliza un circuito electrónico que permite activar los motores en ambos sentidos. Este circuito es comúnmente conocido como puente en H, nombre que recibe por la semejanza que tiene con la letra (figura 4.1).

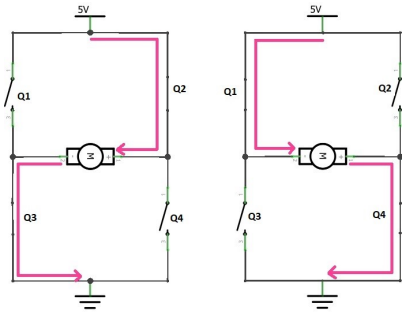


Figura 4.1: Puente H para el control de un motor DC

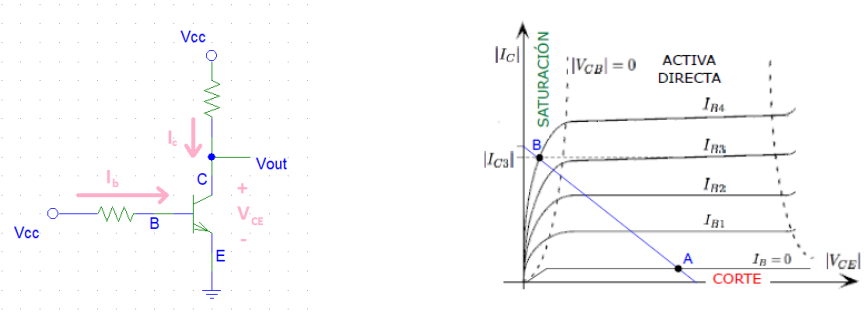
Este circuito se construye, como se ve en la figura, con 4 interruptores. Si se cierran (dejando pasar la corriente) los interruptores Q2 y Q3, dejando abiertos Q1 y Q4, el motor girará en un sentido; al invertir la apertura de los 4 interruptores, el motor girará en sentido contrario.

Q1	Q2	Q3	Q4	F
1	0	1	0	Avance
0	1	0	1	Retroceso
0	0	0	0	Frenado por inercia
1	0	1	0	Frenado rápido
0	1	0	1	Frenado rápido

Tabla 4.1: Tabla de verdad puente H figura 4.1

Nótese, además, que este circuito tiene también un inconveniente: si se cierran los dos interruptores de la misma rama (Q1 y Q3 o Q2y Q4), dejando los otros dos abiertos, el circuito sufriría un cortocircuito de alimentación, situación que debe evitarse [7]. En la tabla 4.1, se ha expuesto toda la casuística del puente en H (tabla de la verdad).

Los interruptores pueden ser mecánicos o electrónicos (transistores). Lo más común es encontrar un circuito de puente-H con interruptores creados con transistores de unión bipolar (BJT), pero también se podrían crear interruptores con transistores de tipo MOSFET. Sin embargo, para aplicaciones de baja potencia, son más adecuados los bipolares.



(a) Esquema BJT

(b) Gráfica de operación

Figura 4.2: Transistor bipolar de unión

Para utilizar un transistor bipolar como interruptor (también conocido como transistor en conmutación), se trabaja en las regiones de corte y saturación (gráfica de la figura 4.2b) siendo en corte donde se encuentra el transistor como interruptor abierto y en saturación como interruptor cerrado (figura 4.3).

Idealmente, en corte la corriente de colector (I_C) es nula y el voltaje entre colector y emisor (V_{CE}) es máximo; y en saturación, la corriente I_C es máxima y el voltaje V_{CE} nulo. Sin embargo, realmente, no se puede obtener un voltaje V_{CE} nulo, y suele ser $V_{CE} \approx$ 0.2V-0.3V.

Para controlar el motor, por tanto, se podría construir con componentes elementales este circuito o bien comprar un circuito integrado. En este caso, y normalmente en los proyectos de robótica, se opta por adquirir un circuito integrado, preparado y optimizado para ellos.

El dispositivo que se va a utilizar para el proyecto es el módulo L298N (figura 4.4) que cuenta con dos puentes en H para el control de dos motores DC (o también podría servir para controlar

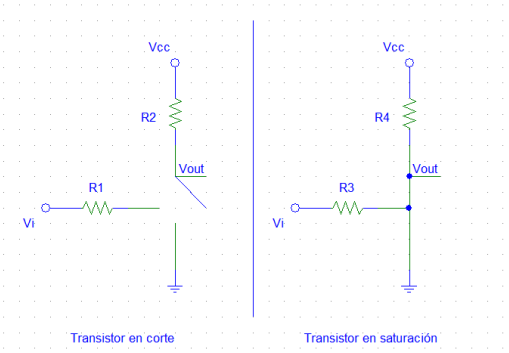
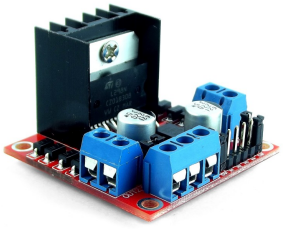
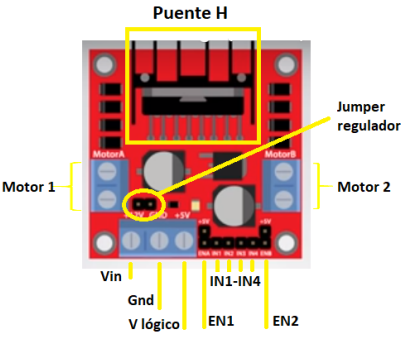


Figura 4.3: Transistor BJT como interruptor

un motor paso a paso); un regulador de tensión LM7805, cuyo uso se puede activar o desactivar; los diodos de protección y una serie de bornes y pines, que se explicarán a continuación, para el conexionado. Se va a hacer uso de este módulo por su pequeño tamaño, bajo coste y la gran cantidad de bibliografía que existe al respecto.



(a) *Driver* L298N



(b) Descripción de los pines *driver* L298N

Figura 4.4: Dipositivo L298N

Como se ha mencionado, dispone de la capacidad de conectar dos motores DC y, para ello, cuenta con dos borneras para conectar sendos polos de los motores. Para el control del sentido de giro de los motores, se hace uso de los 4 pines centrales IN1 e IN2, para el motor 1; e IN3 e IN4 para el motor 2). Los pines EN1 y EN2 permiten habilitar y deshabilitar los motores (retirando el *jumper*) y controlar la velocidad. Estos pines han de conectarse, por tanto, a pines del tipo PWM de la placa Arduino.

Además de los pines mencionados, la tarjeta dispone de unas borneras para la alimentación. En la imagen 4.4b, abajo a la izquierda, se encuentra situada una bornera con 3 entradas: Vin admite tensiones de entre 3V y 35V; a su derecha se encuentra el pin de tierra GND; y, por último, se encuentra la entrada de la tensión lógica.

Para alimentar este dispositivo se puede hacer de dos maneras:

1. Por un lado, utilizando dos fuentes de tensión: en primer lugar, conectando una fuente con la tensión de trabajo del motor por el pin Vin y, en segundo lugar, conectando una fuente de 5V al pin correspondiente. Es importante destacar, que para esta opción hay que desactivar el regulador de tensión (retirando el *jumper*).

2. La segunda manera de alimentar este dispositivo, es activando el *jumper* del regulador de

tensión (es decir, dejar la pieza que une los dos pines conectada) dejando que éste actúe. De esta manera, se necesitará una sola fuente de tensión conectada a Vin. El pin de 5V no necesita ninguna otra fuente de tensión y puede usarse como salida para alimentar algún otro dispositivo, siempre y cuando no se excedan los 500mA de consumo, ya que podría estropearse el dispositivo.

En concreto, para la aplicación de este proyecto se va a alimentar el L298N de la primera forma. De manera, que se retira el *jumper* del regulador de tensión y se conecta la entrada lógica a los 5V del Arduino y una fuente de tensión formada por 4 pilas de 1.5V a Vin.

En este proyecto se van a utilizar dos parejas de motores, cada una de las cuales irá conectada a una bornera del *driver*. Para comprender mejor el funcionamiento de los motores, puede resultar de gran utilidad realizar una tabla de verdad como la tabla 4.2.

	MOTOR 1		**MOTOR 2**	
	IN1	IN2	IN3	IN4
Avanzar	1	0	1	0
Retroceder	0	1	0	1
Giro ida.	1	0	0	1
Giro dcha.	0	1	1	0
Stop	0	0	0	0

Tabla 4.2: Tabla de verdad *driver* L298N

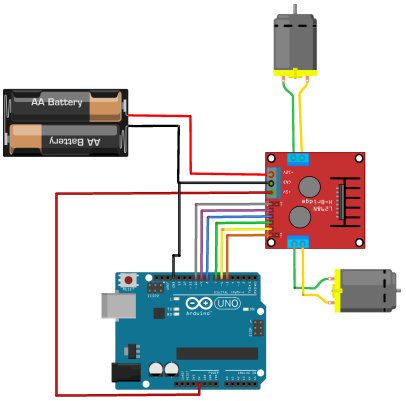


Figura 4.5: Esquema circuito L298N con motores DC

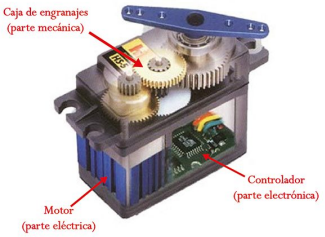
Para el control de los motores, no hay más que implementar la tabla. En la figura 4.5, se ha representado un esquema de conexionado de los motores (cada motor de la figura representa un par de motores, situado uno a cada lado del chasis). Además, en el Anexo A.1.3 se ha adjuntado un *sketch* básico del control de los motores con el *driver* L298N.

4.2. Servomotor

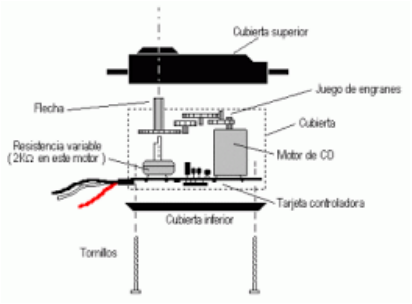
Un servomotor (o simplemente servo) es un motor eléctrico que permite el control de la posición del eje y se diseña de manera que se pueda mover éste un ángulo determinado para mantenerse en él. Son muy utilizados en proyectos de robótica ya que permiten un movimiento controlado y

entregan un mayor torque que los motores DC en solitario, motores incapaces de dar un determinado número de vueltas y detenerse en una posición fija.

Se entiende como servomotor, el conjunto integrado de los componentes electrónicos y mecánicos. Dentro del encapsulado, se encuentra el motor eléctrico encargado de general el movimiento; el sistema de regulación o caja de engranajes, que se encarga de regular la velocidad y el par del motor; el sistema electrónico, encargado de controlar el movimiento del motor mediante el envío de pulsos eléctricos; y, por último, un potenciómetro que se conecta al eje principal y permite conocer en cada momento el ángulo de giro en el que se encuentra.



(a) Estructura servomotor Sg-90



(b) Componentes servomotor

Figura 4.6: Servomotor SG-90

Para el vehículo que se desea montar en este proyecto, se necesita un servomotor que realice y controle el giro del sensor de distancia. El servomotor que se ha elegido es el SG-90 (figura 4.6a), ya que es de pequeño tamaño y peso, e ideal para proyectos en los que se requiere bajo torque. El ángulo de giro de este motor es de 180º, suficiente para esta aplicación.

En la figura 4.7, se puede ver el diagrama de bloques del funcionamiento de un servomotor. Como se puede observar, la señal de entrada debe ser una señal PWM, recibida por el circuito electrónico que la traduce en un movimiento DC. El eje del motor se acopla a un potenciómetro, que forma un divisor de tensión, cuyo voltaje de salida varía según la posición del eje del motor DC. Además, se crea un lazo cerrado con la posición del motor en cada momento, determinada por la posición del potenciómetro, lo que permite realizar el control.

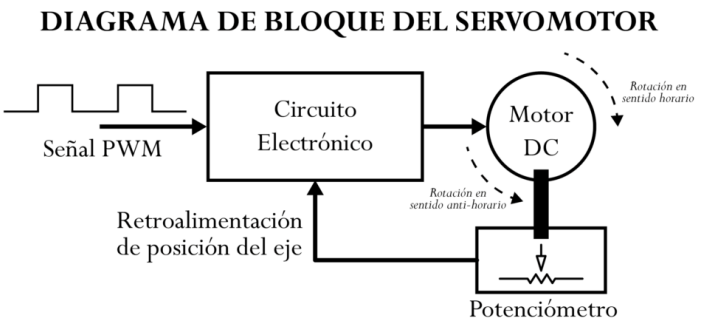


Figura 4.7: Diagrama de bloques, funcionamiento servomotor

Para activar una posición se mandan pulsos, es decir, una señal cuadrada periódica en alto un tiempo determinado. En el caso de este modelo, en el que se puede mover 180º, para pulsos de 1.5 ms el eje se encontrará en el centro (90º), 1 ms en la posición izquierda (0º) y con 2 ms el eje se

colocaría en el extremo derecho (180º).

El dispositivo dispone de 3 cables: la alimentación Vcc (≈4.8V) y GND; y el tercero para el tren de pulsos PWM (SIG). Normalmente, para alimentar este tipo de motores se utiliza una alimentación externa, en vez de utilizar la placa Arduino. En principio, para hacer pequeños proyectos o pruebas de funcionamiento con el Arduino sería suficiente para encender el servomotor; sin embargo, si se va a realizar demasiada fuerza con él o incluso si se pretende alimentar con la misma tarjeta más dispositivos, podría llegar a exceder la capacidad del Arduino dañándolo o haciendo que reinicie. Por lo tanto, para el proyecto se utilizará una fuente de alimentación independiente.

Para utilizar este dispositivo, hace falta incluir en el código la librería <*"Servo.h"*>. El uso del servo es muy sencillo, ya que lo único que hay que hacer es decirle al Arduino qué pin está conectado al tren de pulsos SIG. Además, esta librería incluye una función en la que se puede indicar directamente el ángulo que se desea que gire el servo.

Para comprobar el funcionamiento de este motor, se ha realizado un pequeño programa en Arduino. En la figura 4.8, se muestra el esquema electrónico de conexiones del servomotor al Arduino y en el Anexo A.1.4, se ha incluido el *sketch* del funcionamiento de éste.

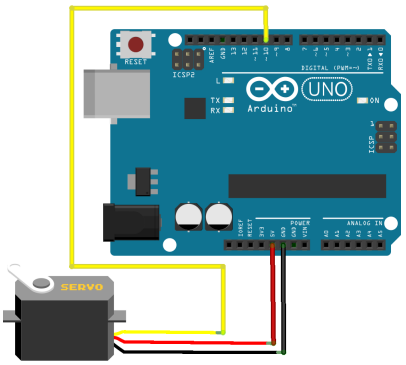


Figura 4.8: Esquema del circuito servomotor

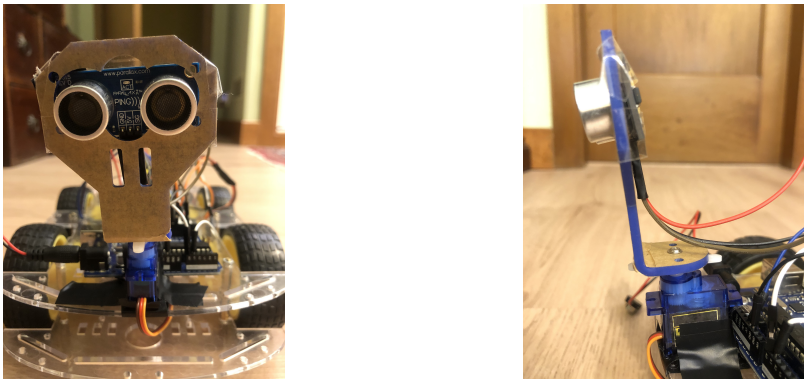
Se utilizará, como se ha mencionado, dicho motor para el control de movimiento del sensor PING. De esta manera, se podrá controlar no solo la distancia a la que se encuentran los obstáculos situados en la dirección de movimiento, sino que se podrá comprobar que no hay ningún obstáculo en diferentes ángulos con respecto a éste. Como se puede observar en la figura 4.9, se coloca el sensor de ultrasonidos PING))) en un cabezal que se encuentra unido al servomotor.

Regulador de tensión

Como se puede observar en la figura 4.8, para hacer la prueba del servomotor se ha conectado éste al Arduino directamente. Sin embargo, cuando se quiere utilizar este componente de manera constante requiere de una alimentación de 5V constantes. Para ello, se hace uso de un regulador de tensión [1].

Un regulador de tensión es un componente electrónico capaz de mantener una tensión constante en su salida, regulando la tensión de entrada. El principio en el que se basa este componente para regular la tensión es la disipación de calor, hecho que se ha de tener en cuenta a la hora de estudiar la idoneidad del dispositivo para cada aplicación ya que toda la energía que se disipa se convierte en calor. La potencia disipada se relaciona con la tensión como dicta la ecuación 4.1.

$$P = (V_{in} - V_{out})I \quad ; \quad [W] \tag{4.1}$$



(a) (b)

Figura 4.9: Servomotor y detector ultrasonidos

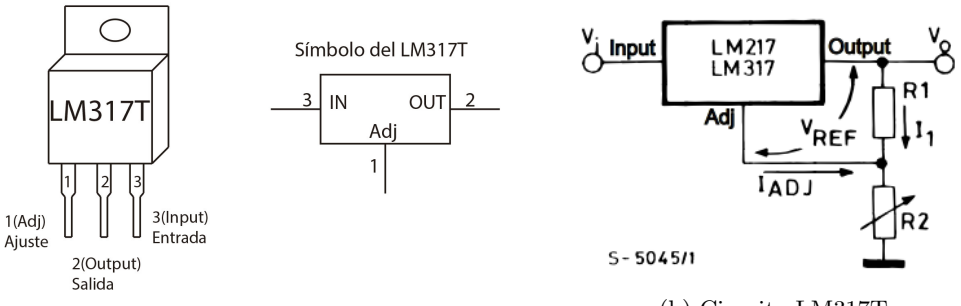
Por lo tanto, si por ejemplo se tiene una tensión de entrada de 9V (V_{in}) y se desea una tensión constante de salida de 5V (V_{out}) para alimentar una carga que consume 1A (I), se estarían convirtiendo 4W de energía eléctrica en calor.

En este caso concreto, se va a utilizar el regulador de tensión LM317T del fabricante *STMicroelectronics*. En la tabla 4.3, se especifican las características electrónicas principales garantizadas por el fabricante. Además, este modelo concreto cuenta con un disipador térmico (chapa plateada), para proteger al componente de sobrecargas por calor.

Tensión de entrada $(V_{min} - V_{max})$V	3 - 4
Tensión de salida $(V_{min} - V_{max})$V	1.2 - 37.0
Corriente máxima salida (A)	1.5
Número de pines	3
Polaridad	Positiva
Tª funcionamiento $(T_{min} - T_{max})$ºC	0 - 125

Tabla 4.3: Tabla características regulador de tensión LM317T

Como se puede ver en la figura 4.10a, cuenta con 3 pines: uno para la tensión de entrada, otro para la tensión de salida deseada y el pin para regular esta última. Para conseguirlo, se ha de montar un pequeño circuito para el que solo se precisa un potenciómetro, conectado a la patilla de Ajuste del dispositivo y una resistencia (figura 4.10b) [4].



(a) Esquema LM317T pines (b) Circuito LM317T

Figura 4.10: Regulador de tensión LM317T

Además, en el datasheet del dispositivo también se puede encontrar otra opción de circuitos

más compleja, que tiene como propósito optimizar el funcionamiento del regulador de tensión, como un capacitor para mejorar la respuesta en transitorio o dos diodos para evitar cortocircuitos en la entrada y en la salida (figura 4.11). En principio, para este proyecto se utilizará la primera propuesta de circuito y se estudiará la posibilidad de usar las mejoras, si se considera oportuno.

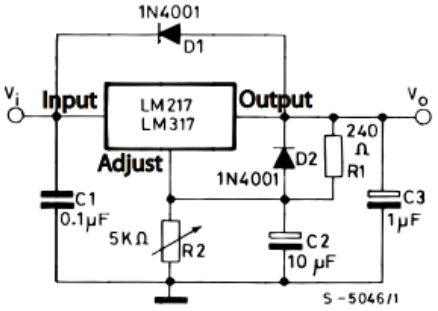


Figura 4.11: Circuito regulador de tensión LM317T optimizado

Capítulo 5

Aplicación desarrollada

Para la realización de la aplicación se construye el vehículo por piezas. Para ello, se ha adquirido un kit de robótica y se incorporarán todos los elementos descritos en los capítulos anteriores. En segundo lugar, haciendo uso de la placa Elegoo UNO se implementarán dos aplicaciones diferentes:

1. La primera aplicación que se desarrollará consiste en un control automático del vehículo. Para ello, se añadirá una distancia "consigna", distancia máxima a la que podrá el vehículo acercarse a los objetos.
2. Para la segunda aplicación se utilizará la pareja de emisión-detección infrarroja para realizar un teleguiado.

5.1. Montaje del vehículo

En primer lugar, se plantea la posibilidad de construir un vehículo con piezas de Lego pero se descarta rápidamente dado su elevado precio. En segundo lugar, se sopesa construir el vehículo comprando las piezas por separado: chasis, motores y ruedas. Pese a que es una opción factible finalmente se descarta ya que existen en el mercado kits de robótica especialmente diseñados para este tipo de proyectos, donde la relación diseño-precio está muy optimizada.

Figura 5.1: Chasis y motores DC

En este caso, se ha escogido un *kit* 4WD (figura 5.1) que incluye para la parte mecánica:

- Dos placas de metacrilato para el chasis
- 4 ruedas de dirección
- 4 motores DC (3-12)V y sus correspondientes sujeciones al chasis
- 4 *encoders* de velocidad

- Un compartimento para 2 pilas recargables de litio (3.7V)
- 1 servomotor
- Tornilloss

En relación a los componentes electrónicos, el *kit* incluye:

- Elegoo UNO
- Sensor de ultrasonidos HC-SR04
- *Shield* o controlador de motores DC o paso a paso

Algunos de los anteriores componentes tuvieron que ser sustituidos por otros más convenientes para la aplicación. Esto, por ejemplo, fue el caso de la *shield* para el controlador de los motores DC. Se trata de un componente que se coloca sobre la placa Arduino, como se puede ver en la figura 5.2, al que se le puede conectar 2 motores DC o bien un motor paso a paso. Este componente cubre todos los pines del Arduino, impidiendo su uso para conectar otros elementos. Es por ello por lo que para el control de los motores DC se hará uso del *driver* L298N explicado en el capítulo 4.1.

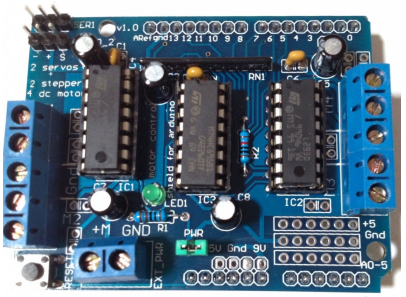


Figura 5.2: Controlador DC incluido en el *kit*

También se decidió sustituir el compartimento de la alimentación ya que las pilas de litio no venían incluidas y su coste es muy elevado. Se optó por comprar otro compartimento para pilas, esta vez para 4 pilas alcalinas AA de 1.5V, consiguiendo una alimentación de 6V.

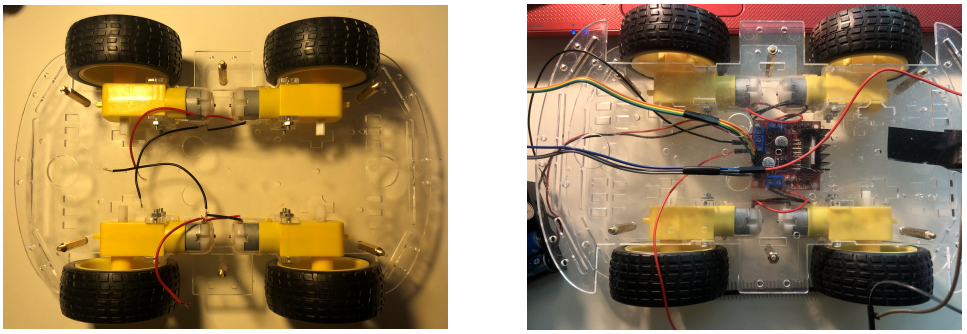
Además de estos últimos componentes electrónicos el vehículo contará con:

- 2 sensores infrarrojos para la detección de suelo
- Un LED rojo
- El circuito del regulador de tensión para el servomotor.

Para su montaje, se conectan los motores por parejas de manera que los dos motores izquierdos y los dos motores derechos estarán cableados en paralelo. Así, como se ha expuesto en el capítulo 4.1, con el mismo *driver* se podrán controlar los 4 motores. Posteriormente, se ajustan al chasis y se colocan las cuatro ruedas (figura 5.3a). Hay que prestar atención a cómo se cablean los motores ya que cada pareja debe estar en sentido opuesto de manera que giren en el mismo sentido cuando se alimenten. Antes de continuar, se comprueba que funcionan correctamente los sentidos de las ruedas.

Una vez realizado este paso, se conectan los motores a la placa L298N encargada de su control: se conectan los dos cables correspondientes a cada pareja en cada bornera de control, se habilitan 6 cables correspondientes a los 4 controles del sentido de giro y a los 2 pines que habilitan los

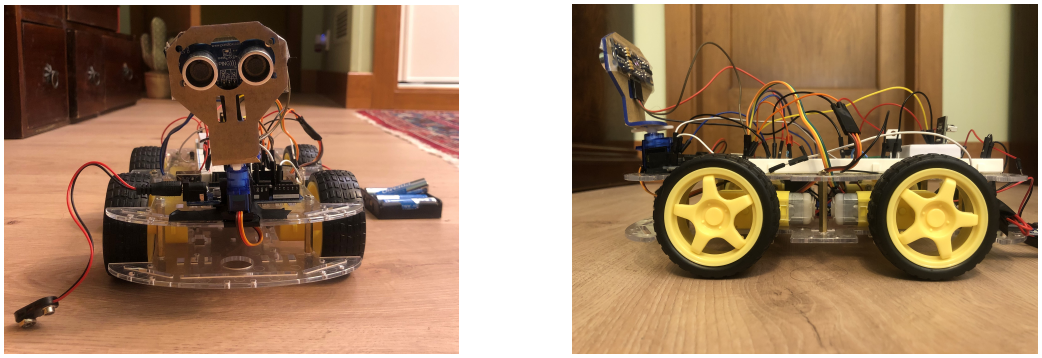
motores. Además, se conecta con cuidado la alimentación (figura 5.3b).



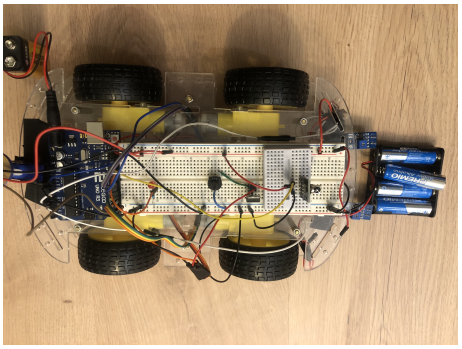
(a) (b)

Figura 5.3: Chasis con motores DC controlados por L298N

Seguidamente, se extraen con cuidado los cables que deberán ir conectados al Arduino y se coloca la parte de arriba del chasis. A continuación, se coloca como se puede ver en la figura 5.4c la protoboard en la que se montan los circuitos del regulador de tensión, el receptor infrarrojo, el LED y todas las conexiones comunes de Tierra y 5V. Finalmente, en la figura 5.4 se pueden observar las tres vistas del vehículo



(a) Alzado (b) Perfil



(c) Planta

Figura 5.4: Vehículo

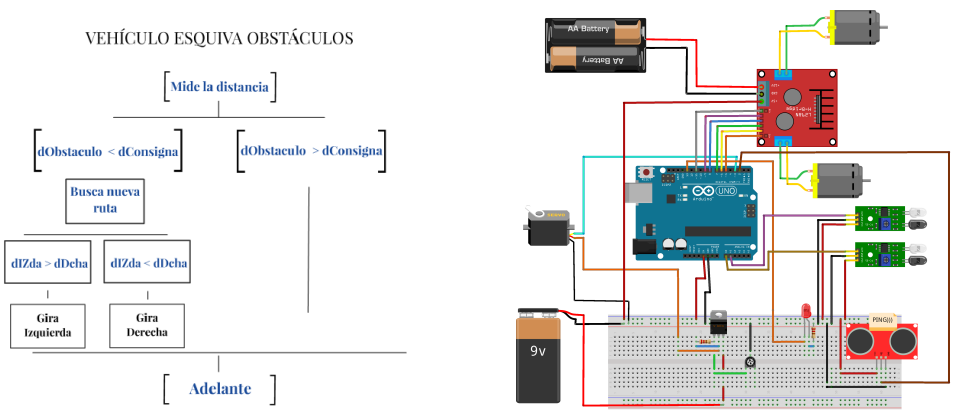
NOTA: Las conexiones de la placa de la figura 5.4 no se corresponden con ninguna de las siguientes aplicaciones.

5.2. Aplicación 1: control automático, vehículo esquiva-obstáculos

Como se ha mencionado, el propósito de la primera aplicación es configurar el Arduino de manera que el vehículo se mueva en "piloto automático" esquivando obstáculos. Para ello, se habrá de introducir una distancia consigna, distancia mínima a la que puede acercarse el vehículo: si la distancia que detecta el sensor es menor, el vehículo buscará una ruta conveniente y cambiará su dirección. La estructura del programa está esquematizada en el diagrama 5.5a y a continuación se van a explicar los pasos del bucle principal:

1. El sensor de ultrasonidos mide la distancia en la dirección del movimiento y, a su vez, los sensores infrarrojos comprueban si hay suelo. Si la distancia detectada es menor (o igual) que la consigna o alguno de los dos detectores infrarrojos detecta que no hay suelo, se encenderá el LED rojo, el vehículo retrocederá y el Arduino buscará una nueva ruta (paso 2). Si nada de esto se cumple el vehículo se mueve hacia delante y el bucle vuelve a empezar.

2. Para encontrar una nueva ruta el servomotor gira a la izquierda (5º) y a la derecha (175º), midiendo en cada posición la distancia a la que el sensor encuentra un objeto. Ambas distancias se comparan y el vehículo gira en la dirección cuyo obstáculo se encuentre más alejado.



(a) Diagrama de flujo

(b) Esquema de conexiones

Figura 5.5: Aplicación 1

A la hora de implementar este programa surgió un problema que no había salido a la luz en ninguna de las pruebas realizadas en los capítulos anteriores. Haciendo uso de los pines tal y como se había hecho para comprobar el funcionamiento de los elementos por separado, el programa no funcionaba; en concreto, de las dos parejas de motores DC que controla el *driver* L298N, solo una de ellas funcionaba. Este problema se solucionaba si se dejaba de usar el servomotor. En un primer momento, se planteó la posibilidad de que el Arduino no fuera capaz de proporcionar corriente suficiente a todos los componentes y se sopesó la posibilidad de comprar una placa con más pines, como es el Arduino Mega.

Sin embargo, antes de hacerlo se encontró la solución en la bibliografía [1]: la librería <*"Servo.h"*> utiliza la interrupción *Timer1*, que hace uso de los pines 9 y 10 de la placa Arduino Uno imposibilitando su uso como pines PWM. Las interrupciones son una manera similar al *delay* de hacer que el Arduino se "congele" un tiempo determinado; es decir, impide realizar cualquier otra función mientras se esté ejecutando ese *delay*. En este caso, el *Timer* que se utiliza es el *Timer1* que, como se ha mencionado, utiliza los pines D9 y D10. Por lo tanto, para esta aplicación se sustituyó el pin 10 por el 11 para el ajuste de la velocidad del motor 2 (EN2)

En el Anexo A.1.5, se ha incluido una captura del *sketch* utilizado para implementar esta aplicación y en la tabla 5.1 se han recogido las conexiones de la placa.

	Arduino	Componente
Digitales	2	Sensor PING)))
	3 (∼)	Servomotor
	5 (∼)	ENA
	6	IN1
	7	IN2
	8	IN3
	9	IN4
	11 (∼)	ENAB
	13	Led Rojo
	A0	Sensor IR (1)
	A1	Sensor IR (2)

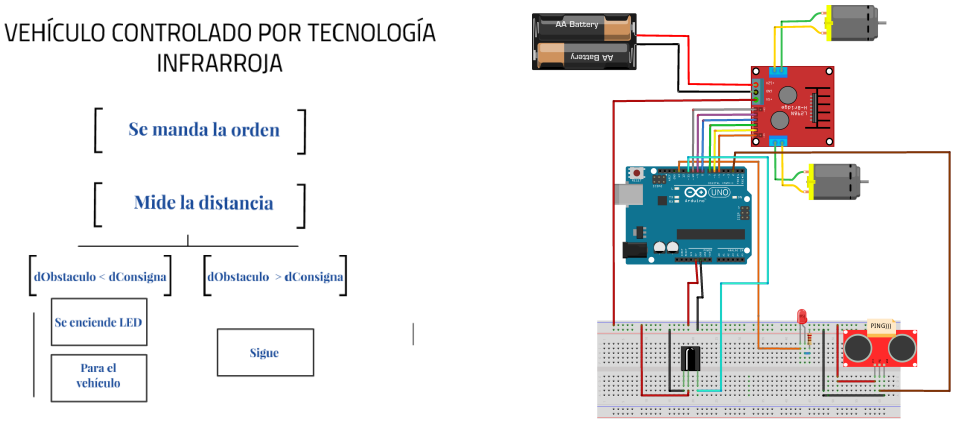
Tabla 5.1: Tabla conexiones aplicación 1

Nótese que, como se ha mencionado en el capítulo 3.2, para el uso de los detectores infrarrojos aunque se haga uso de los pines denotados como analógicos en la placa se declaran como digitales.

En el siguiente **vídeo** se puede comprobar el funcionamiento del vehículo. Como se puede observar, al encontrarse con un obstáculo busca un camino alternativo para continuar su movimiento.

5.3. Aplicación 2: control mediante tecnología inalámbrica

La segunda aplicación que se ha desarrollado es el control de vehículo mediante tecnología infrarroja. Para ello, se hará uso del par emisor-receptor infrarrojo HX1838. Para este caso, se implementará el circuito de la figura 5.6b, cuyo esquema del programa marca la figura 5.6a. En la tabla 5.2 se han recogido las conexiones a la placa Arduino.



(a) Diagrama de flujo

(b) Esquema de conexiones

Figura 5.6: Aplicación 2

El programa que se ha implementado consiste en la utilización del mando y el detector infra-

rrojo, para enviar las órdenes de: adelante, atrás, izquierda, derecha y parar. Además, se añadirá un LED rojo que se encenderá en caso de que el sensor de ultrasonidos detecte que se encuentra un objeto en la dirección de movimiento, se detendrá el vehículo y retrocederá unos centímetros. En el anexo A.1.6, se ha incluido una captura del programa implementado.

	Arduino	Componente
Digitales	2	Sensor PING)))
	5 (∼)	ENA
	6	IN1
	7	IN2
	8	IN3
	9	IN4
	10 (∼)	ENB
	11	Detector IR
	13	Led Rojo

Tabla 5.2: Tabla conexiones aplicación 2

En el siguiente **vídeo** se puede comprobar el funcionamiento de esta aplicación. Al hacerlo, se ha observado un inconveniente principal: como se trata de un módulo IR el direccionamiento del emisor es muy importante, ya que el alcance depende fuertemente del ángulo de emisión, disminuyendo rápidamente al desviarse de la dirección frontal. Esto supone que para controlar el vehículo el mando IR no puede encontrarse muy lejos del vehículo en cuestión.

5.4. Optimización del robot

Una vez estudiados todos los componentes, montado y verificado el pequeño vehículo y diseñadas las dos aplicaciones, se han observado ciertos fallos. Por ello, se proponer algunas mejoras que se podrían realizar para la optimización del proyecto. Dichas mejoras se han agrupado en tres grupos dependiendo de su naturaleza: en primer lugar mejoras electrónicas, grupo que incluye todas aquellas modificaciones que se pueden realizar sobre los componentes electrónicos ya presentes o bien la posibilidad de añadir nuevos que satisfagan carencias observadas; en segundo lugar, mejoras en el programa, aquellos cambios concernientes al *sketch* diseñado en Arduino; y, por último, mejoras relacionadas con el modelo Arduino escogido.

5.4.1. Mejoras electrónicas

En la sección 4.2, se han expuesto las dos posibilidades que el fabricante del regulador de tensión LM317T propone para el circuito que acompaña a este componente. Sin embargo, como se ha expuesto en dicha sección se ha optado por implementar la primera de ellas, ya que era más sencilla y aparentemente válida para este proyecto. La posible mejora que se propone al respecto es la implementación del segundo circuito propuesto por el fabricante (figura 4.11) y la comparación del funcionamiento del regulador en ambos casos. Se prevé que con los diodos de protección con los que cuenta este segundo circuito se proteja de posibles cortocircuitos tanto a la entrada como a la salida del regulador [4].

Por otro lado, otro posible cambio que se le podría añadir es la capacidad de controlarlo con un teléfono móvil, haciendo uso de una *app* dedicada. Para ello, se podría incorporar un módulo *bluetooth*, como podría ser el HC-05 o HC-06. Se trata de unos componentes de muy bajo coste

que incluyen el módulo *bluetooth* como tal y la circuitería necesaria para controlarlo (figura 5.7). La comunicación con la placa Arduino se haría mediante los pines de comunicación serie (RX y TX) y su conexionado sería muy sencillo ya que solo tienen 6 pines, en el caso del módulo HC-05, o 4, en el caso del HC-06. Con este componente y descargando una aplicación para un *Smartphone* (por ejemplo, las aplicaciones *"Bluetooth Terminal HC-05"* o *"LightBlue"*) se podría diseñar un mando para controlar a distancia el movimiento del vehículo, evitando el handicap que supone en la segunda aplicación desarrollada el hecho de que el mando se encuentre alejado del vehículo.

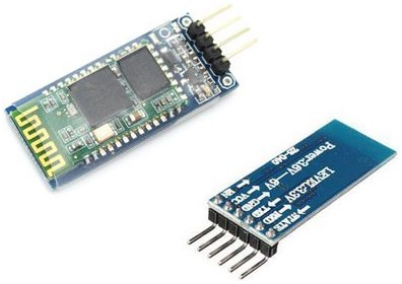


Figura 5.7: Módulo *Bluetooth* HC-05

Otro posibilidad sería añadirle un módulo *wifi* cuyo objetivo sería parecido al de añadirle un módulo *bluetooth* pero cuyo conexionado es algo más complejo.

Para hacerlo más vistoso, podrían haberse añadido, además del ya presente LED rojo, otros dos verde y amarillo, simulando un semáforo y encendiendo unos u otros en función de la distancia a la que se detectase el objeto. Otra alternativa, podría haber sido añadirle un dispositivo LED RGB, capaz de iluminar en diferentes colores dependiendo de la tensión aplicada a cada uno de los tres colores que combina este dispositivo: rojo (**R**ed), verde (**G**reen) y azul (**B**lue).

5.4.2. Mejoras del programa

Para este proyecto concreto únicamente se han diseñado dos programas en los que el vehículo se mueve de manera automática o controlándolo con un mando IR; y se han visto las limitaciones que ambos programas presentan. Sin embargo, muchas son las variaciones que se podrían hacer. Por un lado, se podrían realizar ciertos cambios al propio programa ya implementado, con la finalidad de proporcionarle más robustez, por ejemplo, en la función *BuscarRuta()*, se podría mejorar la medida de distancia a izquierda y derecha aumentando los ángulos, no ciñéndose solo a 2 ángulos por lado. Otro posible cambio, podría ser la optimización de los tiempos de espera entre funciones.

Por otro lado, con este mismo vehículo se podrían implementar programas con diferentes funciones. Por ejemplo, haciendo uso del módulo bluetooth o wifi mencionados en el apartado anterior.

5.4.3. Mejoras del Arduino

Alguna carencia que se ha observado con respecto a la placa Arduino escogida es su número de pines. Para implementar, por ejemplo, la mejora del LED RGB haría falta disponer de 3 pines PWM libres y, sin embargo, con la placa UNO y los componentes que ya posee el robot no sería posible.

Capítulo 6

Conclusiones y trabajo futuro

En cuanto al objetivo principal expuesto en el primer capítulo (control de movimiento de un vehículo apoyado en la placa Arduino), se puede dar por alcanzado, ya que se ha conseguido implementar, por un lado, un control en el cual el vehículo se mueve autónomamente y, por otro lado, un programa para controlar el vehículo remotamente con tecnologías infrarrojas.

Con respecto a los objetivos específicos expuestos en el primer capítulo, también se pueden considerar logrados. Además, podría decirse que el logro de cada uno de ellos acercaba el proyecto a la obtención del objetivo último, permitiendo poco a poco el conocimiento del funcionamiento de cada elemento utilizado.

Además, en el trabajo ha sido necesario el diseño de una metodología para caracterizar sensores de ultrasonidos, cuyos resultados han permitido la elección del dispositivo más adecuado para el proyecto. Asimismo, se ha demostrado que detectores teóricamente iguales proporcionaban resultados muy diferentes, como es el caso de los dispositivos del modelo HC-SR04.

Como se ha mencionado en el capítulo 5.1, ha sido necesario reemplazar ciertos componentes que incluía el *kit* por otros más adecuados para el proyecto. Para ello, ha hecho falta realizar un pequeño estudio de mercado, comparando los diferentes componentes y, en función de su coste, sus características y la bibliografía existente, se han escogido los más adecuados.

Por último, como se ha mencionado en el capítulo anterior, muchas son las mejoras que se pueden incluir al vehículo. Además de las mencionadas, se podría mejorar el aspecto del mismo, "escondiendo" los cables o añadiéndole carrocería.

En este proyecto se han puesto en práctica muchos conocimientos teóricos trabajados a lo largo del grado. Entre ellos cabe mencionar, en primer lugar, la programación. Si bien durante el grado no se ha trabajado específicamente con C++, el haber trabajado con otros lenguajes de programación diferentes ha supuesto que el aprendizaje de uno nuevo se convirtiera en una tarea relativamente sencilla. En segundo lugar, se ha utilizado las teorías de circuitos e instrumentación estudiadas en diferentes materias, para el montaje de los diferentes circuitos y la resolución de los problemas encontrados en el proceso.

En tercer lugar, y al margen de los contenidos de las asignaturas cursadas, para el desarrollo del trabajo se ha necesitado la capacidad de trabajo autónomo que me ha proporcionado el hecho de haber realizado el grado. Si bien he contado con la ayuda de mi tutor para resolver dudas puntuales y para encauzar el esquema del trabajo, dada la situación de los últimos meses toda la comunicación que hemos mantenido ha sido vía e-mail. Por ello, en algunas ocasiones, se he hecho complicado la resolución de los problemas. Un ejemplo claro de ello, fue el problema hallado en la aplicación 1 originado, como se ha explicado anteriormente, por las interrupciones de Arduino de

la librería utilizada. Es muy probable, que de haber podido mantener las reuniones presenciales que veníamos teniendo hasta marzo dicho problema se hubiese solventado con mayor rapidez.

Bibliografía

[1] Neil Cameron. *Arduino Applied Comprehensive Projects for Everyday Electronics-Neil Cameron*. Edinburgh, UK: Apress Media, 2019, pág. 555.

[2] Parallax. Datasheet. *"PING))) Ultrasonic Distance Sensor (28015)"*. `https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf`.

[3] Sparkfun. Datasheet. *Ultrasonic Ranging Module HC - SR04*. `https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf`.

[4] STMicroelectronics. Datasheet. *LM117/217,LM317: 1.2V to 37V voltage regulator*. `https://pdf1.alldatasheet.es/datasheet-pdf/view/22754/STMICROELECTRONICS/LM317T.html`.

[5] Vishay Semiconductors. Datasheet. *Reflective Optical Sensor with Transistor Output*. `https://pdf1.alldatasheet.es/datasheet-pdf/view/252411/VISHAY/TCRT5000.html`.

[6] Brian Evans. *Beggining Arduino programming: writing code for the most popular microcontroller board in the world*. New York: Apress Media, 2011, pág. 270.

[7] John-David Warren, Josh Adams y Harald Molle. *Arduino Robotics*. New York: Apress Media, 2011, pág. 621.

[8] Dale Wheat. *Arduino Internals*. Ed. por Ralph Moore. New York: Apress Media, 2011, pág. 386.

Apéndice A

Anexos

A.1. *Sketchs* Arduino

A.1.1. Caracterización sensores de ultrasonido


```
//HC-SR04 -- 1
int pin_trigger1 = 2;
int pin_echo1 = 3;

//HC-SR04 -- 2
int pin_trigger2 = 8;
int pin_echo2 = 9;

//PING
int pin_PING = 12;


void setup() {
   pinMode(pin_trigger1, OUTPUT);
   pinMode(pin_echo1, INPUT);
   pinMode(pin_trigger2, OUTPUT);
   pinMode(pin_echo2, INPUT);
   Serial.begin (9600);
}
void loop() {
  double cm1 = distanciaHCSR04(pin_trigger1,pin_echo1);
  delayMicroseconds(15);
  double cm2 = distanciaHCSR04(pin_trigger2,pin_echo2);
  delayMicroseconds(15);
  double cm3 = distanciaPING (pin_PING);
  delayMicroseconds(15);
  Serial.print("HC-SR04--1 = ");
  Serial.print(cm1);
  Serial.println();
  Serial.print("HC-SR04--2 = ");
  Serial.print(cm2);
  Serial.println();
  Serial.print("PING))) = ");
  Serial.print(cm3);
  Serial.println();
  delay(3000);
}
```

```
double microsecondsToCentimeter (double microseconds){
  return microseconds /29.0 /2.0;
  }

double distanciaPING (int pin_sensor){
  pinMode(pin_sensor, OUTPUT);
  digitalWrite(pin_sensor,LOW);
  delayMicroseconds(2);
  digitalWrite(pin_sensor,HIGH);
  delayMicroseconds(5);
  digitalWrite(pin_sensor,LOW);
  pinMode(pin_sensor,INPUT);
  double duration = pulseIn(pin_sensor, HIGH);
  return microsecondsToCentimeter(duration);
}
long distanciaHCSR04(int TriggerPin, int EchoPin) {
   long duracion;
   digitalWrite(TriggerPin, LOW);       //para generar un pulso limpio
   delayMicroseconds(4);
   digitalWrite(TriggerPin, HIGH);      //generamos Trigger (disparo) de 10us
   delayMicroseconds(10);
   digitalWrite(TriggerPin, LOW);
   duracion = pulseIn(EchoPin, HIGH);   //tiempo en microsegundos
   return microsecondsToCentimeter (duracion);
}
```

Sensor IR FC-51


```
//EJEMPLO INFRARROJO
int LEDverde = 6; // Led verde
int LEDrojo = 13;  //Led rojo
int IRa = A0; //Entrada digital conectada al sensor infrarrojo

void setup(){
 pinMode(LEDrojo,OUTPUT) ; // Se configuran los LEDs como salidas
 //pinMode(LEDverde,OUTPUT);
 pinMode(IRa,INPUT) ; //Sensor infrarrojo como entrada
 pinMode(IRd,INPUT);
 Serial.begin(9600);
}

void loop(){
 int valorl = digitalRead(IRa) ; //leemos el valor del sensor infrarrojo
 digitalWrite(LEDrojo,(not(valorl)));
 digitalWrite(LEDverde, valor) ;
}
```

A.1.2. Emisor-receptor HX1838

Decodificación botones

decodificarMandoIR §

```
//Author: naylampmechatronics.com

#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup(){
  Serial.begin(9600);
  irrecv.enableIRIn(); // Empezamos la recepción  por IR
}
```

```
void dump(decode_results *results) {
  // Dumps out the decode_results structure. Call this after IRrecv::decode()
  Serial.print("(");
  Serial.print(results->bits, DEC);
  Serial.print(" bits)");

  if (results->decode_type == UNKNOWN) {
    Serial.print("Unknown encoding: ");  }
  else if (results->decode_type == NEC) {
    Serial.print("Decoded NEC: ");  }
  else if (results->decode_type == SONY) {
    Serial.print("Decoded SONY: ");  }
  else if (results->decode_type == RC5) {
    Serial.print("Decoded RC5: ");  }
  else if (results->decode_type == RC6) {
    Serial.print("Decoded RC6: ");  }
  else if (results->decode_type == PANASONIC) {
    Serial.print("Decoded PANASONIC - Address: ");
    Serial.print(results->address, HEX);
    Serial.print(" Value: ");  }
  else if (results->decode_type == LG) {
    Serial.print("Decoded LG ");  }
```

```
  else if (results->decode_type == JVC) {
    Serial.print("Decoded JVC ");  }
  else if (results->decode_type == AIWA_RC_T501) {
    Serial.print("Decoded AIWA RC T501 ");  }
  else if (results->decode_type == WHYNTER) {
    Serial.print("Decoded Whynter ");  }
  Serial.print(results->value, HEX);
  Serial.print(" (HEX) , ");
  Serial.print(results->value, BIN);
  Serial.println(" (BIN)");}

void loop() {
  if (irrecv.decode(&results)) {
    dump(&results);
    irrecv.resume(); // empezamos una nueva recepción
  }
  delay(300);
}
```

Sensor IR para encender 4 LEDs

mandoIR3 §

```
#include "IRremote.h"
int Pinreceptor = 11;

IRrecv receptor(Pinreceptor);
decode_results resultados;
int LEDrojo = 4;
int LEDazul = 5;
int LEDverde = 6;
int LEDamarillo = 7;

void setup(){
  Serial.begin(9600);
  receptor.enableIRIn();
  pinMode(LEDrojo, OUTPUT);
  pinMode(LEDverde, OUTPUT);
  pinMode(LEDazul, OUTPUT);
  pinMode(LEDamarillo, OUTPUT);
}
```

```
void loop(){
  digitalWrite(LEDrojo,LOW);
  digitalWrite(LEDazul,LOW);
  digitalWrite(LEDrojo,LOW);
  digitalWrite(LEDamarillo,LOW);
  if (receptor.decode(&resultados)){
    switch(resultados.value){
      Serial.print(resultados.value);
      case 0x00FF30CF: Serial.println("Tecla: 1");
                       digitalWrite(LEDrojo,HIGH);
                       break;
      case 0x00FF18E7: Serial.println("Tecla: 2");
                       digitalWrite(LEDazul,HIGH);
                       break;
      case 0x00FF7A85: Serial.println("Tecla: 3");
                       digitalWrite(LEDverde,HIGH);
                       break;
      case 0x00FF10EF: Serial.println("Tecla: 4");
                       digitalWrite(LEDamarillo,HIGH);
                       break; }
    receptor.resume();}
  delay(1000);
  }
```

A.1.3. *Driver* L298N para el control de 2 motores DC

motorDC

```
//MOTORES DC, CON DRIVER L298N
//Adjudicación pines del driver:
int IN1 =  9;         // motor izdo 1, amarillo
int IN2 = 8;         // motor izdo 2, verde
int motor1_en = 10;  // enable motor 1, naranja
int IN3 = 7;         // motor dcho 1, azul
int IN4 = 6;         // motor dcho 2, morado
int motor2_en = 5; // enable motor 2, gris

void setup() {
//Declaración del tipo de pine:
 pinMode(IN1,OUTPUT);
 pinMode(IN2,OUTPUT);
 pinMode(motor1_en,OUTPUT);
 pinMode(IN3,OUTPUT);
 pinMode(IN4,OUTPUT);
 pinMode(motor2_en,OUTPUT);
 Serial.begin(9600);
}
void loop() {
  adelante();
  delay(3000);     //Se mantiene 3 segundos hacia delante
  atras();
  delay(3000);
  girodcha();
  delay(3000);
  giroizda();
  delay(3000);
}

//Función de movimiento hacia adelante
void adelante(){
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(motor1_en,200);
  analogWrite(motor2_en,200);}
```

```
void atras(){
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(motor1_en,200);
  analogWrite(motor2_en,200);
}

//Función de giro a la derecha
void girodcha(){
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(motor1_en,150);
  analogWrite(motor2_en,150);
}

//Función de giro a la izquierda
void giroizda(){
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(motor1_en,150);
  analogWrite(motor2_en,150);
}

//Función de stop de los motores
void parar(){
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
}
```

A.1.4. Servomotor

servo

```
//SERVOMOTOR
#include <Servo.h>

Servo servoMotor;              //Declaración de la variable del servomotor
int pin_servo = 10;            //Pin de la placa al que está conectado SIG

void setup() {
  servoMotor.attach(pin_servo);//Se le asigna el pin al servomotor
  Serial.begin(9600);
}

void loop() {
  servoMotor.write(45);      //Se escribe en la variable del servo el ángulo a girar
  delay(500);
  servoMotor.write(90);
  delay(3500);
  servoMotor.write(135);
  delay(500);
  servoMotor.write(90);
  delay(500);
}
```

A.1.5. Aplicación 1

Nota: las funciones de los motores DC y el sensor PING))) utilizadas son las descritas en los apartados A.1.3 y A.1.1.

Servo_PING_DC §

```
//Aplicación 1
#include <Servo.h> //Si quito el servo, el resto de cosas van bien
Servo servoMotor;
int pin_servo = 3;

//Pin PING)))
int pin_sensor = 2;

//Pin LED
int LEDrojo = 13;

//Pines driver L298N
int IN1 = 6;         // motor izdo 1, amarillo
int IN2 = 7;         // motor izdo 2, verde
int motor1_en = 5;   // enable motor 1, naranja
int IN3 = 8;         // motor dcho 1, azul
int IN4 = 9;         // motor dcho 2, morado
int motor2_en = 11; // enable motor 2, gris
//Pines sensores infrarrojos:
int IR1 = A0;
int IR2 = A1;
double dConsigna = 15.00;   //Valor máximo al que se puede encontrar el obstáculo

void setup() {
  //INICIALIZAR SERVO:
  servoMotor.attach (pin_servo);
  //INICILIZAR MOTORES DC:
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(motor1_en,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(motor2_en,OUTPUT);
  //INICIALIZAR LED
  pinMode(LEDrojo,OUTPUT);
  //INICIALIZAR SENSORES IR:
  pinMode(IR1,INPUT);
  pinMode(IR2,INPUT);
  Serial.begin(9600);
}
```

```
void loop() {
  servoMotor.write(90);
  digitalWrite(LEDrojo,LOW);
  adelanteDC();

  double dObstaculo = distancia (pin_sensor);  //Devuelve en la variable dObstacul
  //int valorIR1 = digitalRead(IR1);
  //int valorIR2 = digitalRead(IR2);
  Serial.println(dObstaculo);
  if(dObstaculo < dConsigna){// || valorIR1== LOW || valorIR2 == LOW){
    Serial.println("dentro");
    digitalWrite(LEDrojo,HIGH);
    pararDC();
    calculoRuta();
    }
  if (dObstaculo >= dConsigna){
    adelanteDC();
  }
}
void calculoRuta(){
  atrasDC();
  delay(500);
  pararDC();
  double dIzda = MirarIzda();
  double dDcha =  MirarDcha();
  if (dIzda > dDcha){
    giroIzdaDC();}
  else{
    giroDchaDC();}
    delay(3000);
}

//Funciones del servomotor
double MirarIzda(){
  servoMotor.write(5);
  delay (750);
  double dIzda = distancia (pin_sensor);
  servoMotor.write(90);
  delay(700);
  return dIzda;}



double MirarDcha(){
  servoMotor.write(175);
  delay (750);
  double dDcha = distancia (pin_sensor);
  servoMotor.write(90);
  delay(700);
  return dDcha;}
```

A.1.6. Aplicación 2

Nota: las funciones de los motores DC y el sensor PING))) utilizadas son las descritas en los apartados A.1.3 y A.1.1.


```
//Aplicación 2
#include "IRremote.h"
int Pinreceptor = 11;

IRrecv receptor(Pinreceptor);
decode_results resultados;

//MOTORES DC, CON DRIVER L298N
//Adjudicación pines del driver:
int IN1 =  9;        // motor izdo 1, amarillo
int IN2 = 8;        // motor izdo 2, verde
int motor1_en = 10;  // enable motor 1, naranja
int IN3 = 7;        // motor dcho 1, azul
int IN4 = 6;        // motor dcho 2, morado
int motor2_en = 5; // enable motor 2, gris
int LEDrojo = 13;

//Adjudicación pin sensor US
int sensor = 2;

double dConsigna = 17.00;
void setup() {
  receptor.enableIRIn();
//Declaración del tipo de pines:
 pinMode(IN1,OUTPUT);
 pinMode(IN2,OUTPUT);
 pinMode(motor1_en,OUTPUT);
 pinMode(IN3,OUTPUT);
 pinMode(IN4,OUTPUT);
 pinMode(motor2_en,OUTPUT);
 pinMode(LEDrojo,OUTPUT);
 Serial.begin(9600);
}
```

```
void loop() {
  digitalWrite(LEDrojo,LOW);
  double dObjeto = distancia (sensor);
  Serial.println(dObjeto);
  if (dObjeto < dConsigna){
        parar();
        atras();
        delay(750);
        parar();
        Serial.println("estoy dentro");
        digitalWrite(LEDrojo,HIGH);
        delay(1000);
   }
  if (receptor.decode(&resultados)){
    switch(resultados.value){

      case 0x00FFA25D: Serial.println("Tecla: ON/OFF");
                       digitalWrite(LEDrojo,HIGH);
                       break;
      case 0x00FF629D: Serial.println("Tecla: Vol+");
                       adelante();
                       break;

      case 0x00FF22DD: Serial.println("Tecla: Retroceso");
                       giroizda();
                       break;
      case 0x00FF02FD: Serial.println("Tecla: Play/Pause");
                       parar();
                       break;
      case 0x00FFC23D: Serial.println("Tecla: Adelante");
                       girodcha();
                       break;
      case 0x00FFA857: Serial.println("Tecla: Vol-");
                       atras();
                       break;}
      receptor.resume();
    }
delay(300);
}
```